CHAPTER

14

Administering Networking

IN THIS CHAPTER

Automatically connecting Linux to a network

Using NetworkManager for simple network connectivity

Configuring networking from the command line

Working with network configuration files

Configuring routing, DHCP, DNS, and other networking infrastructure features for the enterprise

Connecting a single desktop system or laptop to a network, particularly one that connects to the Internet, has become so easy that I have put off writing a full chapter on Linux networking until now. If you are trying to connect your Fedora, RHEL, Ubuntu, or another Linux desktop system to the Internet, here's what you can try given an available wired or wireless network interface:

Wired network If your home or office has a wired Ethernet port that provides a path to the Internet and your computer has an Ethernet port, use an Ethernet cable to connect the two ports. After you turn on your computer, boot up Linux and log in. Clicking the NetworkManager icon on the desktop should show you that you are connected to the Internet or allow you to connect with a single click.

Wireless network For a wireless computer running Linux, log in and click the NetworkManager icon on the desktop. From the list of wireless networks that appear, select the one you want and, when prompted, enter the required password. Each time you log in from that computer from the same location, it automatically connects to that wireless network.

If either of those types of network connections works for you, and you are not otherwise curious about how networking works in Linux, that may be all you need to know. However, what if your Linux system doesn't automatically connect to the Internet? What if you want to configure your desktop to talk to a private network at work (VPN)? What if you want to lock down network settings on your server or configure your Linux system to work as a router?

In this chapter, topics related to networking are divided into networks for desktops, servers, and enterprise computing. The general approach to configuring networking in these three types of Linux systems is as follows:

Desktop/laptop networking On desktop or laptop systems, NetworkManager runs by default in order to manage network interfaces. With NetworkManager, you can automatically accept the address and server information that you need to connect to the Internet. However, you can also set address information manually. You can configure things such as proxy servers or virtual private network connections to allow your desktop to work from behind an organization's firewall or to connect through a firewall, respectively.

Server networking Although NetworkManager originally worked best on desktop and laptop network configurations, it now works extremely well on servers. Today, features that are useful for configuring servers, such as Ethernet channel bonding and configuring aliases, can be found in NetworkManager.

Enterprise networking Explaining how to configure networking in a large enterprise can fill several volumes itself. However, to give you a head start using Linux in an enterprise environment, I discuss basic networking technologies, such as DHCP and DNS servers, which make it possible for desktop systems to connect to the Internet automatically and find systems based on names and not just IP addresses.

Configuring Networking for Desktops

Whether you connect to the Internet from Linux, Windows, a smartphone, or any other kind of network-enabled device, certain things must be in place for that connection to work. The computer must have a network interface (wired or wireless), an IP address, an assigned DNS server, and a route to the Internet (identified by a gateway device).

Before I discuss how to change your networking configuration in Linux, let's look at the general activities that occur when Linux is set to connect to the Internet automatically with NetworkManager:

Activate network interfaces NetworkManager looks to see what network interfaces (wired or wireless) are set to start. By default, external interfaces are usually set to start automatically using DHCP, although static names and addresses can be set at install time instead.

Request DHCP service The Linux system acts as a DHCP client to send out a request for DHCP service on each enabled interface. It uses the MAC address of the network interface to identify itself in the request.

Get response from DHCP server A DHCP server, possibly running on a wireless router, cable modem, or other device providing a route to the Internet from your location, responds to the DHCP request. It can provide lots of different types of information to the DHCP client. That information probably contains at least the following:

IP address The DHCP server typically has a range of Internet Protocol (IP) addresses that it can hand out to any system on the network that requests an address. In more secure environments, or one in which you want to be sure that specific machines get specific addresses, the DHCP server provides a specific IP address to requests from specific MAC addresses. (MAC addresses are made to be unique among all network interface cards and are assigned by the manufacturer of each card.)

Subnet mask When the DHCP client is assigned an IP address, the accompanying subnet mask tells that client which part of the IP address identifies the subnetwork and which identifies the host. For example, an IP address of 192.168.0.100 and subnet mask of 255.255.255.0 tell the client that the network is 192.168.0 and the host part is 100.

Lease time When an IP address is dynamically allocated to the DHCP client (Linux system), that client is assigned a lease time. The client doesn't own that address but must lease it again when the time expires and request it once again when the network interface restarts. Usually, the DHCP server remembers the client and assigns the same address when the system starts up again or asks to renew the lease. The default lease time is typically 86,400 seconds (24 hours) for IPV4 addresses. The more plentiful IPV6 addresses are assigned for 2,592,000 seconds (30 days) by default.

Domain name server Because computers like to think in numbers (such as IP addresses like 192.168.0.100) and people tend to think in names (such as the hostname `www.example.com`), computers need a way to translate hostnames into IP addresses and sometimes the opposite as well. The domain name system (DNS) was designed to handle that problem by providing a hierarchy of servers to do name-to-address mapping on the Internet. The location of one or more DNS servers (usually two or three) is usually assigned to the DHCP client from the DHCP host.

Default gateway Although the Internet has one unique namespace, it is actually organized as a series of interconnected subnetworks. In order for a network request to leave your local network, it must know what node on your network provides a route to addresses outside of your local network. The DHCP server usually provides the "default gateway" IP address. By having network interfaces on both your subnetwork and the next network on the way to the ultimate destination of your communication, a gateway can route your packets to their destination.

Other information A DHCP server can be configured to provide all kinds of information to help the DHCP client. For example, it can provide the location of an NTP server (to sync time between clients), font server (to get fonts for your X display), IRC server (for online chats), or print server (to designate available printers).

Update local network settings After the settings are received from the DHCP server, they are implemented as appropriate on the local Linux system. For example, the IP address is set on the network interface, the DNS server entries are added to the local /etc/resolv.conf file (by NetworkManager), and the lease time is stored by the local system so it knows when to request that the lease be renewed.

All of the steps just described typically happen without your having to do anything but turn on your Linux system and log in. Now suppose that you want to be able to verify your network interfaces or change some of those settings. You can do that using the tools described in the next sections.

Checking your network interfaces

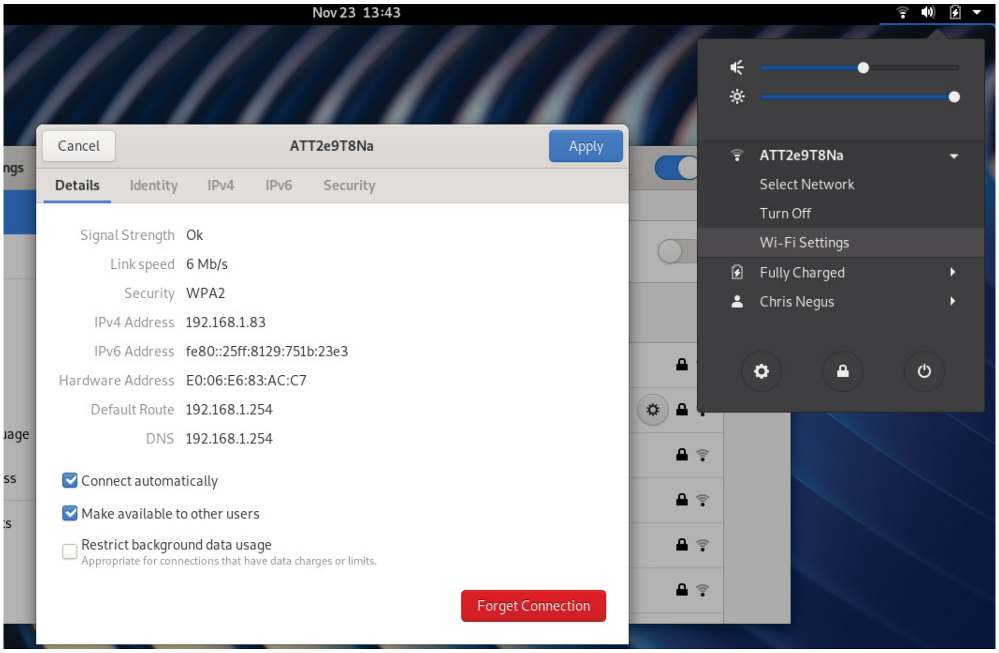
There are both graphical and command-line tools for viewing information about your network interfaces in Linux. From the desktop, NetworkManager tools and the Cockpit web user interface are good places to start.

Checking your network from NetworkManager

The easiest way to check the basic setting for a network interface is to open the pull-down menu at the upper-right corner of your desktop and select your active network interface. Figure 14.1 shows the Wi-Fi settings for an active network on a Fedora GNOME 3 desktop.

FIGURE 14.1

Checking network interfaces with NetworkManager



Signal Strength Ok
Link speed 6 Mb/s
Security WPA2
IPv4 Address 192.168.1.83
IPv6 Address fe80::25ff:8129:751b:23e3
Hardware Address E0:06:E6:83:AC:C7
Default Route 192.168.1.254
DNS 192.168.1.254

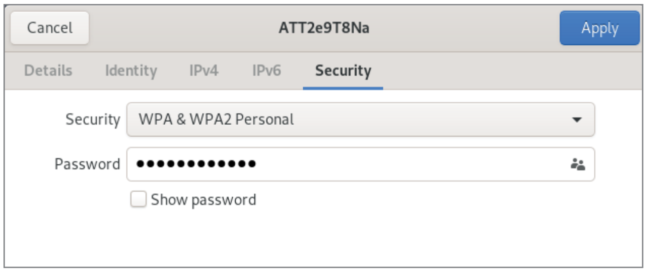
☑ Connect automatically
☑ Make available to other users
☐ Restrict background data usage
Appropriate for connections that have data charges or limits.

As you can see in Figure 14.1, both IPv4 and IPv6 addresses are assigned to the interface. The IP address 192.168.1.254 offers both a DNS service and a route to external networks.

To see more about how your Linux system is configured, click one of the tabs at the top of the window. For example, Figure 14.2 shows the Security tab, where you can select the type of security connection to the network and set the password needed to connect to that network.

FIGURE 14.2

Viewing network settings with NetworkManager



Security: WPA & WPA2 Personal

Password: ●●●●●●●●●●●●●

Checking your network from Cockpit

Provided you have enabled Cockpit, you can see and change information about your network interfaces through your web browser. On your local system, you open `https://localhost:9090/network` to go directly to the Cockpit Networking page for your local system. Figure 14.3 shows an example of this.

From the Cockpit Networking page, you can see information about all of your network interfaces at once. In this case, there are three network interfaces: `wlp2s0` (the active wireless interface), `enp4s0` (an inactive wired interface), and `virbr0` (an inactive interface into the network connected to any virtual machines running on the local system).

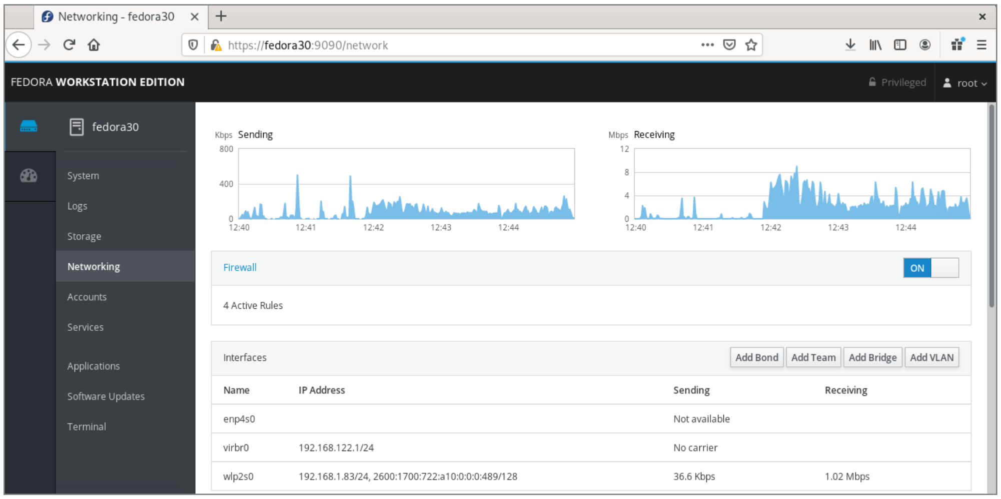
At the top of the Cockpit Networking page, you can see data being sent from and received on the local system. Select a network interface to see a page displaying activities for that particular interface.

Select Firewall to see the list of services that are allowed into your system. For example, Figure 14.4 shows that UDP ports are open for three services (DHCPv6 client, Multicast DNS, and Samba Client). DHCPv6 lets the system acquire an IPv6 address from the network. Multicast DNS and Samba Client services can allow the autodetection of printers, share file systems, and a variety of devices and shared resources.

The only TCP service shown here as open is ssh. With the ssh service (TCP port 22) open, the sshd service running on your local system is available for remote users to log into your local system.

FIGURE 14.3

Viewing and changing network settings from Cockpit



Firewall ON

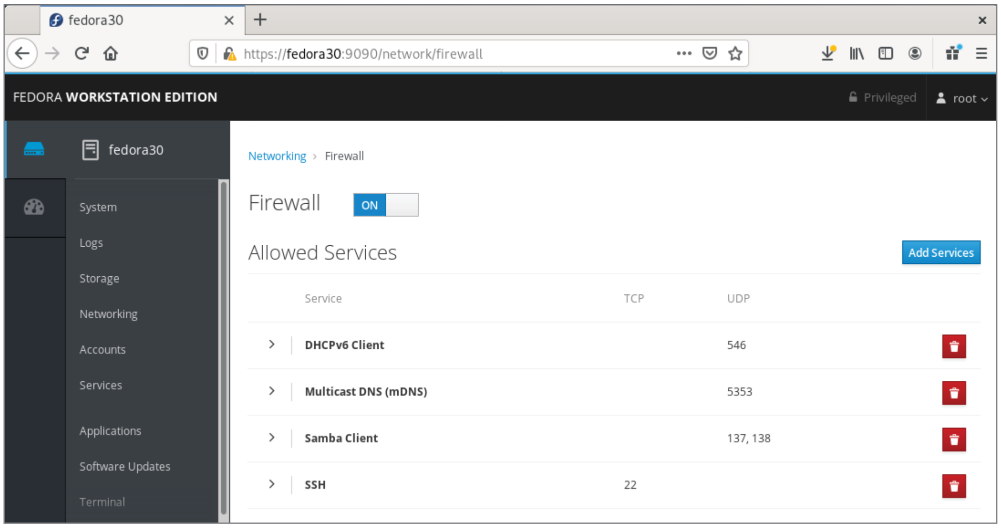
4 Active Rules

Interfaces

Name	IP Address	Sending	Receiving
enp4s0		Not available	
virbr0	192.168.122.1/24	No carrier	
wlp2s0	192.168.1.83/24, 2600:1700:722:a10:0:0:0:489/128	36.6 Kbps	1.02 Mbps

FIGURE 14.4

View services that are accessible through the firewall from Cockpit



Firewall ON

Allowed Services

Service	TCP	UDP
DHCPv6 Client		546
Multicast DNS (mDNS)		5353
Samba Client		137, 138
SSH	22	

The fact that those ports are open doesn't necessarily mean that the services are running. However, if they are running, the computer's firewall will allow access to them.

More advanced features available from the Cockpit Networking page allow you to add bonds, teams, bridges, and VLANs to your local network interfaces.

Checking your network from the command line

To get more detailed information about your network interfaces, try running some commands. There are commands that can show you information about your network interfaces, routes, hosts, and traffic on the network.

Viewing network interfaces

To see information about each network interface on your local Linux system, enter the following:

```
# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue
        state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp4s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500
        qdisc fq_codel state DOWN group default qlen 1000
    link/ether 30:85:a9:04:9b:f9 brd ff:ff:ff:ff:ff:ff
3: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500
        qdisc mq state UP group default qlen 1000
    link/ether e0:06:e6:83:ac:c7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.83/24 brd 192.168.1.255 scope global
        dynamic noprefixroute wlp2s0
       valid_lft 78738sec preferred_lft 78738sec
    inet6 2600:1700:722:a10::489/128 scope global dynamic
noprefixroute
       valid_lft 5529sec preferred_lft 5229sec
    inet6 fe80::25ff:8129:751b:23e3/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500
       qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:0c:69:0a brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
       valid_lft forever preferred_lft forever
```

The `ip addr show` output displays information about your network interfaces, in this case from a laptop running Fedora 30. The `lo` entry in the first line of the output shows the loopback interface, which is used to allow network commands run on the local system to connect to the local system. The IP address for localhost is 127.0.0.1/8 (the /8 is CIDR notation, indicating that 127.0 is the network number and 0.1 is the host number). Add a -s option (ip -s addr show) to see statistics of packet transmissions and errors associated with each interface.

In this case, the wired Ethernet interface (enp4s0) is down (no cable), but the wireless interface is up (wlp2s0). The MAC address on the wireless interface (wlp2s0) is e0:06:e6:83:ac:c7 and the Internet (IPv4) address is 192.168.1.83. An IPv6 address is also enabled.

Older versions of Linux are used to assign more generic network interface names, such as eth0 and wlan0. Now interfaces are named by their locations on the computer's bus. For example, the first port on the network card seated in the third PCI bus for a Fedora system is named p3p1. The first embedded Ethernet port would be em1. Wireless interfaces sometimes appear using the name of the wireless network as the device name.

Another popular command for seeing network interface information is the ifconfig command. By default, ifconfig shows similar information to that of ip addr, but ifconfig also shows the number of packets received (RX) and transmitted (TX) by default, as well as the amount of data and any errors or dropped packets:

```
# ifconfig wlp2s0
wlp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.83  netmask 255.255.255.0
             broadcast 192.168.1.255
        inet6 2600:1700:722:a10:b55a:fca6:790d:6aa6
             prefixlen 64  scopeid 0x0<global>
        inet6 fe80::25ff:8129:751b:23e3
             prefixlen 64  scopeid 0x20<link>
        inet6 2600:1700:722:a10::489
             prefixlen 128  scopeid 0x0<global>
        ether e0:06:e6:83:ac:c7  txqueuelen 1000  (Ethernet)
        RX packets 208402  bytes 250962570 (239.3 MiB)
        RX errors 0  dropped 4  overruns 0  frame 0
        TX packets 113589  bytes 13240384 (12.6 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
Checking connectivity to remote systems
```

To make sure that you can reach systems that are available on the network, you can use the ping command. As long as the computer responds to ping requests (not all do), you can use ping to send packets to that system in a way that asks them to respond. Here is an example:

```
$ ping host1
PING host1 (192.168.1.15 ) 56(84) bytes of data.
64 bytes from host1 (192.168.1.15 ): icmp_seq=1 ttl=64 time=0.062 ms
64 bytes from host1 (192.168.1.15 ): icmp_seq=2 ttl=64 time=0.044 ms
^C
```

```
--- host1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1822ms
rtt min/avg/max/mdev = 0.044/0.053/0.062/0.009 ms
```

The `ping` command shown here continuously pings the host named `host1`. After a few pings, press Ctrl+C to end the pings, and the last few lines show you how many of the `ping` requests succeeded.

You could have used the IP address (192.168.1.15, in this case) to see that you could reach the system. However, using the hostname gives you the additional advantage of knowing that your name-to-IP-address translation (being done by your DNS server or local hosts file) is working properly as well. In this case, however, `host1` appeared in the local `/etc/hosts` file.

Checking routing information

Routing is the next thing that you can check with respect to your network interfaces. The following snippets show you how to use the `ip` and `route` commands to do that:

```
# ip route show
default via 192.168.122.1 dev ens3 proto dhcp metric 20100
192.168.122.0/24 dev ens3 proto kernel scope link src 192.168.122.194
metric 100
```

The `ip route show` command example illustrates that the 192.168.122.1 address provides the route to the host from a RHEL 8 VM over the `ens3` network interface. Communications to any address in the 192.168.122.0/24 range from the VM (192.168.122.194) goes over that interface. The `route` command can provide similar information:

```
# route
Kernel IP routing table
Destination   Gateway    Genmask       Flags Metric Ref Use Iface
default       homeportal 0.0.0.0       UG    600    0     0 wlp2s0
192.168.1.0   0.0.0.0    255.255.255.0 U     600    0     0 wlp2s0
192.168.122.0 0.0.0.0    255.255.255.0 U     0      0     0 virbr0
```

The output from the kernel routing table is from a Fedora system with a single active external network interface. The wireless network interface card is on PCI slot 2, port 1 (`wlp2`). Any packets destined for the 192.168.1 network use the `wlp2` NIC. Packets destined for any other location are forwarded to the gateway system at 192.168.1.0. That system represents my router to the Internet. Here's a more complex routing table:

```
# route
Kernel IP routing table
Destination   Gateway      Genmask         Flags Metric Ref Use Iface
default       gateway      0.0.0.0         UG    600    0     0 wlp3s0
10.0.0.0      vpn.example. 255.0.0.0       U     50     0     0 tun0
10.10.135.0   0.0.0.0      255.255.217.0   U     50     0     0 tun0
vpn.example.  gateway      255.255.255.255 UGH   600    0     0 wlp3s0
```

```
172.17.0.0    0.0.0.0      255.255.0.0     U     0      0      0 docker0
192.168.1.0   *            255.255.255.0   U     600    0      0 wlp3s0
```

In the route example just shown, there is a wireless interface (`wlp3s0`) as well as an interface representing a virtual private network (VPN) tunnel. A VPN provides a way to have encrypted, private communications between a client and a remote network over an insecure network (such as the Internet). Here the tunnel goes from the local system over the `wlan0` interface to a host named `vpn.example.com` (some of the name is truncated).

All communication to the 192.168.1.0/24 network still goes directly over the wireless LAN. However, packets destined for the 10.10.135.0/24 and 10.0.0.0/8 networks are routed directly to `vpn.example.com` for communication with hosts on the other side of the VPN connection over the tunneled interface (`tun0`).

A special route is set up for communications to containers (`docker0`) running on the local system on network 172.17.0.0. All other packets go to the default route via the address 192.168.1.0. As for the flags shown in the output, a `U` says the route is up, a `G` identifies the interface as a gateway, and an `H` says the target is a host (as is the case with the VPN connection).

So far, I have shown you the routes to leave the local system. If you want to follow the entire route to a host from beginning to end, you can use the `traceroute` command (`dnf install traceroute`). For example, to trace the route a packet takes from your local system to the `google.com` site, type the following `traceroute` command:

```
# traceroute google.com
traceroute to google.com (74.125.235.136), 30 hops max, 60 byte pkts
...
 7  rrcs-70-62-95-197.midsouth.biz.rr.com (70.62.95.197)  ...
 8  ge-2-1-0.rlghncpop-rtr1.southeast.rr.com (24.93.73.62)  ...
 9  ae-3-0.cr0.dca10.tbone.rr.com (66.109.6.80) ...
10  107.14.19.133 (107.14.19.133)  13.662  ms  ...
11  74.125.49.181 (74.125.49.181)  13.912  ms ...
12  209.85.252.80 (209.85.252.80)  61.265  ms ...
13  66.249.95.149 (66.249.95.149)  18.308  ms ...
14  66.249.94.22 (66.249.94.22)  18.344  ms ...
15  72.14.239.83 (72.14.239.83)  85.342  ms ...
16  64.233.174.177 (64.233.174.177)  167.827  ms ...
17  209.85.255.35 (209.85.255.35)  169.995  ms  ...
18  209.85.241.129 (209.85.241.129)  170.322  ms  ...
19  nrt19s11-in-f8.1e100.net (74.125.235.136)  169.360ms  ...
```

I truncated some of the output to drop off some of the initial routes and the amount of time (in milliseconds) that the packets were taking to traverse each route. Using `traceroute`, you can see where the bottlenecks are along the way if your network communication is stalling.

Viewing the host and domain names

To see the hostname assigned to the local system, type `hostname`. To just see the domain portion of that name, use the `dnsdomainname` command:

```
# hostname
spike.example.com
# dnsdomainname
example.com
```

Configuring network interfaces

If you don't want to have your network interfaces assigned automatically from a DHCP server (or if there is no DHCP server), you can configure network interfaces manually. This can include assigning IP addresses, the locations of DNS servers and gateway machines, and routes. This basic information can be set up using NetworkManager.

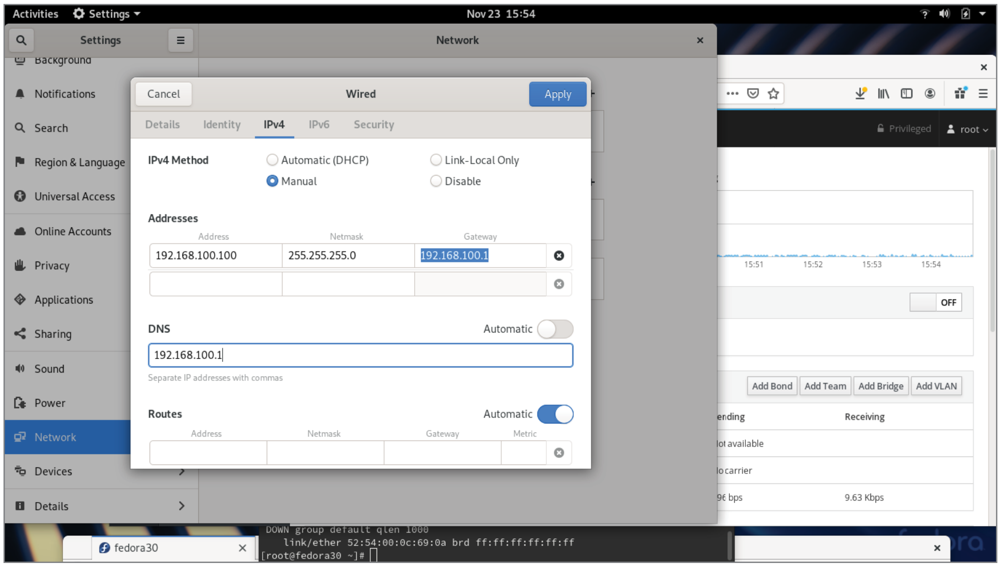
Setting IP addresses manually

To change the network configuration for your wired network interface through Network-Manager, do the following:

1. Select the Settings icon from the drop-down menu in the upper-right corner of the desktop and select Network.
2. Assuming that you have a wired NIC that is not yet in use, select the settings button (small gear icon) next to the interface that you want to change.
3. Choose IPv4 and change the IPv4 Method setting from Automatic (DHCP) to Manual.
4. Fill in the following information (only Address and Netmask are required):
   a. **Address**: The IP address that you want to assign to your local network interface. For example, 192.168.100.100.
   b. **Netmask**: The subnetwork mask that defines which part of the IP address represents the network and which part identifies the host. For example, a netmask of 255.255.255.0 would identify the network portion of the previous address as 192.168.100 and the host portion as 100.
   c. **Gateway**: The IP address of the computer or device on the network that acts as the default route. The default route will route packets from the local network to any address that is not available on the local network or via some other custom route.
   d. **DNS servers**: Fill in the IP addresses for the system providing DNS service to your computer. If there is more than one DNS server, add the others in a comma-separated list of servers.
5. Click the Apply button. The new information is saved, and the network is restarted using the new information. Figure 14.5 shows an example of those network settings.

FIGURE 14.5

Changing network settings with NetworkManager



IPv4 Method: Automatic (DHCP) [✓ Manual] Link-Local Only Disable

Addresses

Address	Netmask	Gateway
192.168.100.100	255.255.255.0	192.168.100.1

DNS — Automatic: off

192.168.100.1

Separate IP addresses with commas

Routes — Automatic: on

Address	Netmask	Gateway	Metric

```
DOWN group default qlen 1000
    link/ether 52:54:00:0c:69:0a brd ff:ff:ff:ff:ff:ff
[root@fedora30 ~]#
```

Setting IP address aliases

You can attach multiple IP addresses to a single network interface. In the same Network-Manager screen, this is done by simply filling in a subsequent Addresses box and adding the new IP address information. Here are a few things you should know about adding address aliases:

- A netmask is required for each address, but a gateway is not required.
- The Apply button stays grayed out until you include valid information in the fields.
- The new address does not have to be on the same subnetwork as the original address, although it is listening for traffic on the same physical network.

After adding the address 192.168.100.103 to my wired interface, running `ip addr show enp4s0` displays the following indication of the two IP addresses on the interface:

```
2: enp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
state UP group default qlen 1000
    link/ether 30:85:a9:04:9b:f9 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.100/24 brd 192.168.100.255 scope
          global noprefixroute enp4s0
       valid_lft forever preferred_lft forever
    inet 192.168.100.103/24 brd 192.168.100.255 scope
          global secondary noprefixroute enp4s0
       valid_lft forever preferred_lft forever
```

For information on setting up aliases directly in configuration files, refer to the section "Setting alias network interfaces" later in this chapter.

Setting routes

When you request a connection to an IP address, your system looks through your routing table to determine the path on which to connect to that address. Information is sent in the form of packets. A packet is routed in the following different ways, depending on its destination:

- The local system is sent to the `lo` interface.
- A system on your local network is directed through your NIC directly to the intended recipient system's NIC.
- Any other system is sent to the gateway (router) that directs the packet on to its intended address on the Internet.

Of course, what I have just described here is one of the simplest cases. You may, in fact, have multiple NICs with multiple interfaces to different networks. You may also have multiple routers on your local network that provide access to other private networks. For example, suppose you have a router (or other system acting as a router) on your local network; you can add a custom route to that router via NetworkManager. Using the NetworkManager example shown previously, scroll down the page to view the Routes section. Then add the following information:

Address The network address of the subnetwork you route to. For example, if the router (gateway) will provide you access to all systems on the 192.168.200 network, add the address 192.168.200.0.

Netmask Add the netmask needed to identify the subnetwork. For example, if the router provides access to the Class C address 192.168.200, you could use the netmask 255.255.255.0.

Gateway Add the IP address for the router (gateway) that provides access to the new route. For example, if the router has an IP address on your 192.168.1 network of 192.168.1.199, add that address in this field.

Click Apply to apply the new routing information. You may have to restart the interface for this to take effect (for example, `ifup enp4s0`). Enter `route -n` to make sure the new routing information has been applied.

```
# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref Use Iface
0.0.0.0         192.168.100.1   0.0.0.0         UG    1024   0     0 p4p1
192.168.100.0   0.0.0.0         255.255.255.0 U     0      0     0 p4p1

192.168.200.0   192.168.1.199   255.255.255.0 UG    1      0     0 p4p1
```

In the example just shown, you can see that the default gateway is 192.168.100.1. However, any packets destined for the 192.168.200 network are routed through the gateway host at IP address 192.168.1.199. Presumably that host has a network interface that faces the 192.168.200 network, and it is set up to allow other hosts to route through it to that network.

See the section "Setting custom routes" later in this chapter for information on how to set routes directly in configuration files.

Configuring a network proxy connection

If your desktop system is running behind a corporate firewall, you might not have direct access to the Internet. Instead, you might have to reach the Internet via a proxy server. Instead of allowing you full access to the Internet, a proxy server lets you make requests only for certain services outside of the local network. The proxy server then passes those requests on to the Internet or another network.

Proxy servers typically provide access to web servers (`http://` and `https://`) and FTP servers (`ftp://`). However, a proxy server that supports SOCKS can provide a proxy service for different protocols outside of the local network. (*SOCKS* is a network protocol made to allow client computers to access the Internet through a firewall.) You can identify a proxy server in NetworkManager and have communications for selected protocols go through that server (from the Settings window, select Network and then select Network Proxy).

Instead of identifying a proxy server to your network interfaces (via NetworkManager), you can configure your browser to use a proxy server directly by changing your Firefox preferences to use a proxy server. Here's how to define a proxy server from the Firefox window:

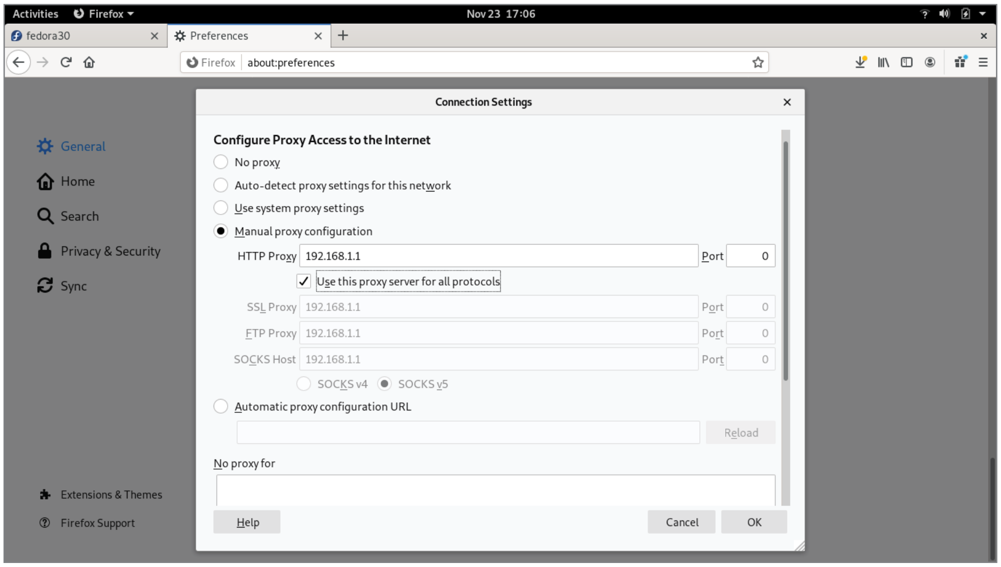
1. From Firefox, select Preferences. The Firefox Preferences window appears.
2. From the Firefox Preferences window, scroll down to Network Settings and select Settings.
3. From the Connection Settings window that appears, you can try to autodetect the proxy settings or, if you set the proxy in NetworkManager, you can choose to use system proxy settings. You can also select Manual Proxy Configuration, fill in the following information, and click OK.
   a. **HTTP Proxy**: The IP address of the computer providing the proxy service. This causes all requests for web pages (`http://` protocol) to be forwarded to the proxy server.
   b. **Port**: The port associated with the proxy service. By default, the port number is 3128, but it can differ.
   c. **Use this proxy server for all protocols**: Select this box to use the same proxy server and port associated with the HTTP proxy for all other service requests. This causes other proxy settings to be grayed out. (Instead of selecting this box, you can set those proxy services separately.)

d. **No Proxy for**: Add the hostname or IP address for any system that you want to be able to contact with Firefox directly without going through the proxy server. You don't need to add localhost and the local IP address (127.0.0.1) in this box, since those addresses are already set not to redirect.

Figure 14.6 shows an example of the Configure Proxy Access to the Internet window filled in to configure a connection to a proxy server located at IP address 192.168.1.1 for all protocols. After you click OK, all requests from the Firefox browser to locations outside of the local system are directed to the proxy server, which forwards those requests on to the appropriate server.

FIGURE 14.6

Setting up Firefox to use a proxy server



Configure Proxy Access to the Internet

- No proxy
- Auto-detect proxy settings for this network
- Use system proxy settings
- ● Manual proxy configuration

HTTP Proxy 192.168.1.1 Port 0

✓ Use this proxy server for all protocols

SSL Proxy 192.168.1.1 Port 0

FTP Proxy 192.168.1.1 Port 0

SOCKS Host 192.168.1.1 Port 0

SOCKS v4 ● SOCKS v5

- Automatic proxy configuration URL

No proxy for

Configuring Networking from the Command Line

While NetworkManager does a great job of autodetecting wired networks or presenting you with lists of wireless networks, sometimes you need to abandon the NetworkManager GUI and commands or Cockpit to configure the features that you need. These are some of the networking features in RHEL and Fedora described in the coming sections:

Basic configuration: See how to use `nmtui` to configure basic networking with a menu-based interface from a shell. This tool provides an intuitive interface for configuring networking on servers that have no graphical interface for running GUI-based tools.

Configuration files: Understand configuration files associated with Linux networking and how to configure them directly.

Ethernet channel bonding: Set up Ethernet channel bonding (multiple network cards listening on the same IP address).

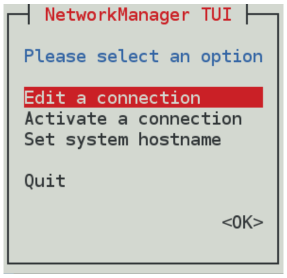
Configure networking with nmtui

Many servers don't have graphical interfaces available. So, if you want to configure networking, you must be able to do so from the shell. One way to do that is to edit networking configuration files directly. Another method is to use menu-based commands that let you press arrow and Tab keys to navigate and forms you fill in to configure your network interface.

The `nmtui` command (`yum install NetworkManager-tui`) provides a menu-based interface that runs in the shell. As root, enter **`nmtui`** to see a screen similar to the one presented in Figure 14.7.

FIGURE 14.7

Configuring networking with NetworkManager TUI



```
NetworkManager TUI

Please select an option

Edit a connection
Activate a connection
Set system hostname

Quit

                  <OK>
```

Use arrow keys and the Tab key to move around the interface. With the item you want to select highlighted, press Enter to select it. The interface is limited to modifying the following kinds of information: Edit or Activate a connection (network interface cards) and Set system hostname (hostname and DNS configuration).

Editing a NetworkManager TUI connection

From the NetworkManager TUI screen displayed, here is how to edit an existing connection.

1. **Edit a connection**: With "Edit a connection" highlighted, press Enter. A list of network devices (usually wired or wireless Ethernet cards) is displayed, along with any wireless networks to which you have connected in the past.
2. **Network devices**: Highlight one of the network devices (in my case, I chose a wired Ethernet interface) and press Enter.

3. **IPv4 Configuration**: Move to the IPv4 Configuration show button and press Enter. The Edit Connection window that appears lets you change information relating to the selected network device.
4. **Change to Manual**: You can leave the Profile Name and Device fields as they are. By default, Automatic is enabled. Automatic is what allows the network interface to come up automatically on the network if a DHCP service is available. To enter address and other information yourself, use the Tab key to highlight the Automatic field and press the spacebar; then use the arrow keys to highlight Manual and press Enter.
5. **Addresses**: Now fill in the address information (IP address and netmask). For example, 192.168.0.150/24 (where 24 is the CIDR equivalent for the 255.255.255.0 netmask).
6. **Gateway**: Type in the IP address for the computer or router that is supplying the route to the Internet.
7. **DNS servers**: Type in the IP addresses of either one or two DNS servers to tell the system where to go to translate hostnames you request into IP addresses.
8. **Search domains**: The Search domains entries are used when you request a host from an application without using a fully qualified domain name. For example, if you type `ping host1` with an `example.com` search path, the command would try to send ping packets to `host1.example.com`.
9. **Routing**: You can set custom routes by highlighting Edit in the Routing field and pressing Enter. Fill in the Destination/Prefix and Next Hop fields and select OK to save the new custom route.
10. **Other selections**: Of the other selections on the screen, consider setting "Never use this network for default route" if the network doesn't connect to wider networks and "Ignore automatically obtained routes" if you don't want those features to be set automatically from the network. Figure 14.8 shows the screen after Manual has been selected and the address information has been filled in.

Tab to the OK button and press the spacebar. Then click Quit to exit.

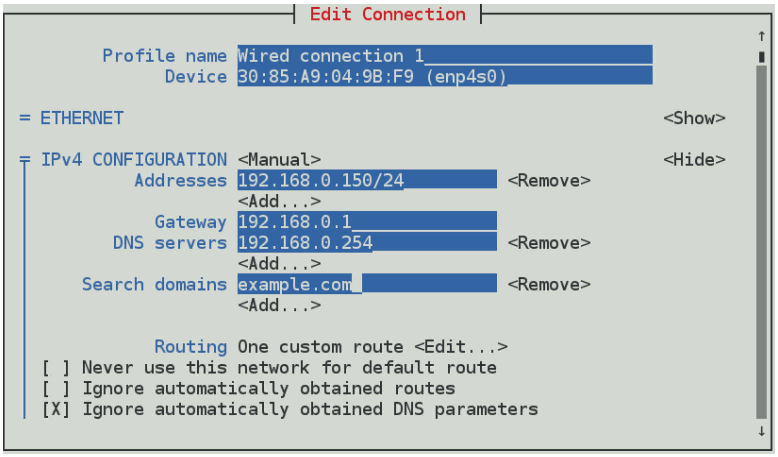
Understanding networking configuration files

Whether you change your network setup using NetworkManager or `nmtui`, most of the same configuration files are updated. In Fedora and RHEL, network interfaces and custom routes are set in files in the `/etc/sysconfig/network-scripts` directory.

Open the `/usr/share/doc/initscripts/sysconfig.txt` file for descriptions of network-scripts configuration files (available from the `initscripts` package).

FIGURE 14.8

Set static IP addresses by selecting Manual from the Edit Connection screen.



```
                         ┤ Edit Connection ├

         Profile name Wired connection 1_______________________
               Device 30:85:A9:04:9B:F9 (enp4s0)_______________

= ETHERNET                                                    <Show>

╤ IPv4 CONFIGURATION <Manual>                                 <Hide>
│          Addresses 192.168.0.150/24_________ <Remove>
│                    <Add...>
│            Gateway 192.168.0.1______________
│        DNS servers 192.168.0.254____________ <Remove>
│                    <Add...>
│     Search domains example.com______________ <Remove>
│                    <Add...>
│
│            Routing One custom route <Edit...>
│ [ ] Never use this network for default route
│ [ ] Ignore automatically obtained routes
│ [X] Ignore automatically obtained DNS parameters
```

One thing to be careful about is that NetworkManager believes that it controls the files in the network-scripts directory. So keep in mind that if you set manual addresses on an interface that NetworkManager has configured for DHCP, it could overwrite changes that you made manually to the file.

Network interface files

Configuration files for each wired, wireless, ISDN, dialup, or other type of network interface are represented by files in the `/etc/sysconfig/network-scripts` directory that begin with `ifcfg-`*`interface`*. Note that *`interface`* is replaced by the name of the network interface.

Given a network interface for a wired NIC as `enp4s0`, here's an example of an `ifcfg-enp4s0` file for that interface, configured to use DHCP:

```
DEVICE=enp4s0
TYPE=Ethernet
BOOTPROTO=dhcp
ONBOOT=yes
DEFROUTE=yes
UUID=f16259c2-f350-4d78-a539-604c3f95998c
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
```

```
NAME="System enp4s0"
PEERDNS=yes
PEERROUTES=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
```

In this `ifcfg-enp4s0` example, the first two lines set the device name and the type of interface to `Ethernet`. The `BOOTPROTO` variable is set to `dhcp`, which causes it to request address information from a DHCP server. With `ONBOOT=yes`, the interface starts automatically at system boot time. IPV6 settings say to initialize IPV6 and use the IPV6 settings that are presented, but the interface will continue to initialize if there is no IPV6 network available. Other settings say to use peer DNS automatically and route values that are detected.

Here's what a simple `ifcfg-enp4s1` file might look like for a wired Ethernet interface that uses static IP addresses:

```
DEVICE=enp4s1
HWADDR=00:1B:21:0A:E8:5E
TYPE=Ethernet
BOOTPROTO=none
ONBOOT=yes
USERCTL=no
IPADDR=192.168.0.140
NETMASK=255.255.255.0
GATEWAY=192.168.0.1
```

In this `ifcfg-enp4s1` example, because this is setting the address and other information statically, `BOOTPROTO` is set to `none`. Other differences are needed to set the address information that is normally gathered from a DHCP server. In this case, the IP address is set to 192.168.0.140 with a netmask of 255.255.255.0. The `GATEWAY=192.168.0.1` identifies the address of the router to the Internet.

Here are a couple of other settings that might interest you:

`PEERDNS`: Setting `PEERDNS=no` prevents DHCP from overwriting the `/etc/resolv.conf` file. This allows you to set which DNS servers your system uses without fear of that information being erased by data that is provided by the DHCP server.

`DNS`***?***: If an `ifcfg` file is being managed by NetworkManager, it sets the address of DNS servers using DNS*?* entries. For example, `DNS1=192.168.0.2` causes that IP address to be written to `/etc/resolv.conf` as the first DNS server being used on the system. You can have multiple `DNS`***?*** entries (`DNS2=`, `DNS3=`, and so on).

In addition to configuring the primary network interfaces, you can also create files in the `/etc/sysconfig/network-scripts` directory that can be used to set aliases (multiple IP addresses for the same interface), bonded interfaces (multiple NICs listening on the same address), and custom routes. Those are described later in this chapter.

Other networking files

In addition to the network interface files, there are other network configuration files that you can edit directly to configure Linux networking. Here are some of those files.

/etc/sysconfig/network file

System-wide settings associated with your local networking can be included in your /etc/sysconfig/network file. The system's hostname was commonly set in this file up to RHEL 6, but other settings can be added to this file as well. Here is an example of the contents of an /etc/sysconfig/network file:

```
GATEWAY=192.168.0.1
```

In the previous example, the default GATEWAY is set to 192.168.0.1. Different interfaces can use different GATEWAY addresses. For other settings that can appear in the network files, check the sysconfig.txt file in the /usr/share/doc/initscripts directory.

/etc/hostname file

In RHEL and Fedora releases, the system's hostname is stored in the /etc/hostname file. For example, if the file included the hostname host1.example.com, that hostname would be set each time the system booted up. You can check how the current hostname is set at any time by typing the hostname command.

/etc/hosts file

Before DNS was created, translating hostnames to IP addresses was done by passing around a single hosts file. While there were only a few dozen and then a few hundred hosts on the Internet, this approach worked pretty well. But as the Internet grew, the single hosts file became unscalable and DNS was invented.

The /etc/hosts file still exists on Linux systems. It can still be used to map IP addresses to hostnames. The /etc/hosts file is a way to set up names and addresses for a small local network or just create aliases in order to make it easier to access the systems that you use all the time.

Here's an example of an /etc/hosts file:

```
127.0.0.1  localhost localhost.localdomain
::1        localhost localhost.localdomain
192.168.0.201  node1.example.com node1 joe
192.168.0.202  node2.example.com node2 sally
```

The first two lines (127.0.0.1 and ::1) set addresses for the local system. The IPv4 address for the local host is 127.0.0.1; the IPv6 address for the local host is ::1. There are also entries for two IP addresses. You could reach the first IP address (192.168.0.201) by the names node1.example.com, node1, or joe. For example, typing ping joe results in packets being sent to 192.168.0.201.

/etc/resolv.conf file

DNS servers and search domains are set in the /etc/resolv.conf file. If NetworkManager is enabled and running, you should not edit this file directly. Using DNS?= entries from ifcfg-* files, NetworkManager overwrites the /etc/resolv.conf file so that you would lose any entries you add to that file. Here's an example of the /etc/resolv.conf file that was modified by NetworkManager:

```
# Generated by NetworkManager
nameserver 192.168.0.2
nameserver 192.168.0.3
```

Each nameserver entry identifies the IP address of a DNS server. The order defines the order in which the DNS servers are checked. It's normal to have two or three nameserver entries, in case the first is not available. More than that, and it can take too long for an unresolvable hostname to get checked for each server.

Another type of entry that you can add to this file is a search entry. A *search entry* lets you indicate domains to be searched when a hostname is requested by its base name instead of its entire fully qualified domain name. You can have multiple search entries by identifying one or more domain names after the search keyword, as in this example:

```
search example.com example.org example.net
```

The search options are separated by spaces or tabs.

/etc/nsswitch.conf

Unlike in earlier releases, the /etc/nsswitch.conf file is managed by the authselect command and should not be modified manually. To make changes, edit the /etc/authselect/user-nsswitch.conf file and run authselect apply-changes.

Settings in the /etc/nsswitch.conf file determine that hostname resolution is done by first searching the local /etc/hosts file (files) and then DNS servers listed in the /etc/resolv.conf file (dns). The myhostname value is used to ensure that an address is always returned for the host. This is how the hosts entry in the /etc/resolv.conf file appears in Red Hat Enterprise Linux:

```
hosts:      files dns myhostname
```

You can add other locations, such as Network Information Service (nis or nisplus) databases, for querying hostname-to-IP-address resolution. You can also change the order in which the different services are queried. You can check that hostname-to-IP-address resolution is working properly using different commands.

If you want to check that your DNS servers are being queried properly, you can use the host or dig commands, as in, for example:

```
$ host redhat.com
redhat.com has address 209.132.183.105
redhat.com mail is handled by 10 us-smtp-inbound-1.mimecast.com.
redhat.com mail is handled by 10 us-smtp-inbound-2.mimecast.com.
$ dig redhat.com
; <<>> DiG 9.11.11-RedHat-9.11.11-1.fc30 <<>> redhat.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9948
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;redhat.com.          IN  A
...
;; ANSWER SECTION:
redhat.com.     3600  IN  A  09.132.183.105
;; Query time: 49 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sat Nov 23 19:16:14 EST 2019
```

By default, the `host` command produces simpler output for DNS queries. It shows the IP address for `redhat.com` and the names of the mail servers (MX records) that serve `redhat.com`. The `dig` command shows information similar to what appears in the files that hold DNS records. The `QUESTION SECTION` part of the output shows that the address section asked for the address of `redhat.com` and the `ANSWER SECTION` part showed the answer (209.132.183.105). You can also see the address of the DNS server that was queried.

The `host` and `dig` commands are only used to query DNS servers. They don't check the `nsswitch.conf` file to find other places to query, such as the local hosts file. For that, you would have to use the `getent` command:

```
# getent hosts node1
192.168.0.201  node1
```

This `getent` example finds a host named `node1` that was entered into my local `/etc/hosts` file. The `getent` command can be used to query any information setup in the `nsswitch.conf` file. For example, typing `getent passwd root` shows the entry for the root user account in the local file, but it can also query a remote LDAP database for user information if you have configured that feature, as described in Chapter 11, "Managing User Accounts."

Setting alias network interfaces

Sometimes you might want your network interface card listening on multiple IP addresses. For example, if you were setting up a web server that was serving secure content (`https`) for multiple domains (`example.com`, `example.org`, and so on), each domain would require a separate IP address (associated with a separate certificate). In that case, instead of adding multiple network interface cards to the computer, you could simply create multiple aliases on a single NIC.

To create an alias network interface in RHEL 6 and earlier Fedora releases, you just have to create another `ifcfg-` file. Following the example of an `eth0` interface on a RHEL system, you could create an eth0:0 interface associated with the same network interface card. To do this, create a file in the /etc/sysconfig/network-scripts directory called ifcfg-eth0:0 that contains information such as the following:

```
DEVICE=eth0:0
ONPARENT=yes
IPADDR=192.168.0.141
NETMASK=255.255.255.0
```

The example code creates an alias for the network interface eth0 called eth0:0. Instead of ONBOOT, the ONPARENT entry says to bring up this interface if the parent (eth0) is started and listen on address 192.168.0.141. You can bring up that interface by typing ifup eth0:0. You can then check that the interface came up using the ip command:

```
$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
       pfifo_fast state UP qlen 1000
    link/ether f0:de:f1:28:46:d9 brd ff:ff:ff:ff:ff:ffinet
    192.168.0.140/24 brd 192.168.0.255 scope global
    eth0inet 192.168.0.141/24 brd 192.168.0.255 scope global secondary
    eth0:0inet6 fe80::f2de:f1ff:fe28:46d9/64 scope link
       valid_lft forever preferred_lft forever
```

You can see that the network interface card represented by eth0 is listening on two addresses: 192.168.0.140 (eth0) and 192.168.0.141 (eth0:0). So, this system will respond to packets destined for either of those two addresses. You could add more IP addresses to that interface by creating more ifcfg-eth0:? files (ifcfg-eth0:1, ifcfg-eth0:2, and so on).

In more recent RHEL and Fedora systems, you can create aliases directly in the primary ifcfg file for an alias. For example, a primary (192.168.0.187) and alias (192.168.99.1) address for a NIC interface named p4p1 might be represented by the following address settings in the ifcfg-p4p1 file:

```
IPADDR=192.168.0.187
PREFIX=24
IPADDR1=192.168.99.1
PREFIX1=24
```

Setting up Ethernet channel bonding

Ethernet channel bonding allows you to have more than one network interface card on a computer associated with a single IP address. There are several reasons you might want to do this:

High availability Multiple NICs on the same IP address can ensure that if one subnet goes down or one NIC breaks, the address can still be reached on a NIC connected to another subnet.

Performance If there is too much network traffic to be handled by one NIC, you can spread that traffic across multiple NICs.

In Red Hat Enterprise Linux and Fedora on a computer with multiple NICs, you can set up Ethernet channel bonding by creating a few ifcfg files and loading the necessary module. You can start with one bonding file (for example, ifcfg-bond0) and then point multiple ifcfg-eth*?* files at that bond interface. Then you can load the bond module.

Depending on the type of bonding that you want to do, you can set your bonding interface to different modes. Using the BONDING_OPTS variable, you define the mode and other bonding options (all of which are passed to the bonding module). You can read about the bonding module by entering modinfo bonding or by installing the kernel-docs package and reading the bonding.txt file from the /usr/share/doc/kernel-doc*/Documentation/networking/directory.

Here is an example of the file that defines a bonded interface. The file in this example is /etc/sysconfig/network-scripts/ifcfg-bond0:

```
DEVICE=bond0
ONBOOT=yes
IPADDR=192.168.0.50
NETMASK=255.255.255.0
BOOTPROTO=none
BONDING_OPTS="mode=active-backup"
```

The bond0 interface in this example uses the IP address 192.168.0.50. It starts up on boot. The BONDING _ OPTS sets the bonding mode to active-backup. This means that only one NIC is active at a time, and the next NIC only takes over when the previous one fails (failover). No network interface card is associated with the bond0 interface yet. For that, you must create separate ifcfg file options. For example, create an /etc/sysconfig/network-scripts/ifcfg-eth0 that looks like the following (then create eth1, eth2, eth3, and so on for each NIC that you want to use in the bonding interface):

```
DEVICE=eth0
MASTER=bond0
SLAVE=yes
BOOTPROTO=none
ONBOOT=yes
```

With the eth0 interface used as part of the bond0 interface, there is no IP address assigned. That's because the eth0 interface uses the IP address from the bond0 interface by defining itself as a slave (SLAVE=yes) to bond0 (MASTER=bond0).

The last thing that you want to do is to make sure that the bond0 interface is set to use the bonding module. To do that, create an /etc/modprobe.d/bonding.conf file that contains the following entry:

```
alias bond0 bonding
```

Because all of the interfaces are set to ONBOOT=yes, the bond0 interface starts and all of the *eth?* interfaces are available as they are needed.

Setting custom routes

On a simple network configuration, communications that are destined for the local network are directed to the appropriate interface on your LAN, while communications for hosts outside of your LAN go to a default gateway to be sent on to remote hosts. As an alternative, you can set custom routes to provide alternative paths to specific networks.

To set a custom route in Fedora and RHEL, you create a configuration file in the `/etc/sysconfig/network-scripts` directory. In that route, you define the following:

- ***GATEWAY?*** The IP address of the node on the local network that provides the route to the subnetwork represented by the static route.
- ***ADDRESS?*** The IP address representing the network that can be reached by the static route.
- ***NETMASK?*** The netmask that determines which part of the *ADDRESS?* represents the network and which represents the hosts that can be reached on that network.

The name of each custom route file is *route-interface*. So, for example, a custom route that can be reached through your `eth0` interface would be named `route-eth0`. You could have multiple custom routes in that file, with each route entry replacing the *?* with the interface number:

```
ADDRESS0=192.168.99.0
NETMASK0=255.255.255.0
GATEWAY0=192.168.0.5
```

In this example, any packet destined for a host on the 192.168.99 network would be sent through the local `eth0` interface and directed to the gateway node at 192.168.0.5. Presumably, that node would provide a route to another network containing hosts in the 192.168.99 address range. This route would take effect when the `eth0` network interface was restarted.

To check that the route is working after you restart the network interface, you could type the following:

```
# route
Kernel IP routing table
Destination  Gateway     Genmask         Flags Metric Ref Use Iface
default      192.168.0.1 0.0.0.0         UG    0      0     0 eth0
192.168.0.0  *           255.255.255.0   U     1      0     0 eth0
192.168.99.0 192.168.0.5 255.255.255.0   UG    0      0     0 eth0
```

The output from the `route -n` command shows that the default route (anything not destined for the local network 192.168.0 or the 192.168.99 network) is via the 192.168.0.1 address. Any packets destined for the 192.168.99 network are directed through the address 192.168.0.5.

If you wanted to add more custom routes, you could add them to this same `route-eth0` file. The next set of information would be named `ADDRESS1`, `NETMASK1`, `GATEWAY1`, and so on.

Configuring Networking in the Enterprise

So far, the network configuration described in this chapter has centered on setting up single systems to connect to a network. Features available in Linux can go well beyond that by providing software that supports the actual network infrastructure needed by host computers to communicate.

The following sections introduce you to a few of the network infrastructure types of services available in Linux. Full implementation of these features is beyond the scope of this book, but know that if you find yourself needing to manage network infrastructure features, the following sections will give you a sense of how those features are implemented in Linux.

Configuring Linux as a router

If you have more than one network interface on a computer (typically two or more NICs), you can configure Linux as a router. To make this happen, all that is needed is a change to one kernel parameter that allows packet forwarding. To turn on the `ip_forward` parameter immediately and temporarily, enter the following as root:

```
# cat /proc/sys/net/ipv4/ip_forward
0
# echo 1 > /proc/sys/net/ipv4/ip_forward
# cat /proc/sys/net/ipv4/ip_forward
1
```

Packet forwarding (routing) is disabled by default, with the value of `ip_forward` set to 0. By setting it to `1`, packet forwarding is immediately enabled. To make this change permanent, you must add that value to the `/etc/sysctl.conf` file, so that it appears as follows:

```
net.ipv4.ip_forward = 1
```

With that file modified as shown, each time the system reboots, the value for `ip_forward` is reset to `1`. (Notice that `net.ipv4.ip _ forward` reflects the actual location of the `ip _ forward` file, minus `/proc/sys`, and with dots replacing slashes. You can change any kernel parameters set in the `/proc/sys` directory structure in this way.)

When a Linux system is used as a router, it is often used as a firewall between a private network and a public network, such as the Internet. If that is the case, you might also want to use that same system as a firewall that does network address translation (NAT) and provides DHCP service, so the systems on the private network can route through the Linux system using private IP addresses. (See Chapter 25, "Securing Linux on a Network," for information on working with Linux firewall rules using the `iptables` facility.)

Configuring Linux as a DHCP server

Not only can a Linux system use a DHCP server to get its IP address and other information, it can also be configured to act as a DHCP server itself. In its most basic form, a DHCP server can hand out IP addresses from a pool of addresses to any system that requests an IP address. Usually, however, the DHCP server also distributes the locations of DNS servers and the default gateway.

Configuring a DHCP server is not something that should be done without some thought. Don't add a DHCP server on a network that is not under your control and that already has a working DHCP server. Many clients are set up to get address information from any DHCP server that will hand it out.

DHCP service is provided by the `dhcp-server` package in Fedora and RHEL. The service is named `dhcpd`. The primary configuration file is `/etc/dhcp/dhcpd.conf` for IPv4 networks (there is a `dhcpd6.conf` file in the same directory to provide DHCP service for IPv6 networks). By default, the `dhcpd` daemon listens on UDP port 67, so remember to keep that port open on your firewall.

To configure a DHCP server, you could copy the `dhcpd.conf.example` file from the `/usr/share/doc/dhcp-server` directory and replace the `/etc/dhcp/dhcpd.conf` file. Then modify it as you like. Before using that file, however, you want to change the domain-name options to reflect your domain and IP address ranges to suit those you are using. The comments in the file will help you do this.

When you install some virtualization and cloud services on a Linux system, a DHCP server is set up by default for you within that system. For example, when you install KVM and start the `libvirtd` service in RHEL or Fedora, it automatically configures a default private network in the 192.168.122.0/24 address range. When you launch virtual machines, they are given IP addresses in that range. When you install and start the Docker service on those Linux distributions, it likewise sets up a private network and hands out IP addresses to Docker containers launched on that system.

Configuring Linux as a DNS server

In Linux, most professional Domain Name System (DNS) servers are implemented using the Berkeley Internet Name Domain (BIND) service. This is implemented in Fedora and RHEL by installing the `bind`, `bind-utils`, and `bind-libs` packages. For added security, some people install the `bind-chroot` package.

By default, `bind` is configured by editing the `/etc/named.conf` file. Hostname-to-IP-address mapping is done in zone files located in the `/var/named` directory. If you install the `bind-chroot` package, `bind` configuration files are moved under the `/var/named/chroot` directory, which attempts to replicate the files from `/etc` and `/var` that are needed to configure `bind` so that the named daemon (which provides the service) is confined to the `/etc/named/chroot` directory structure.

If you are interested in trying out `bind`, I recommend that you first try it out by configuring DNS for a small home network behind a firewall as a way to make it easier for the people in your household to communicate with each other. You can lock down the IP addresses of the machines in your home by attaching MAC addresses of each computer's network interface card to specific IP addresses on a DHCP server and then mapping those names to addresses in a DNS server.

Caution

Before you create a public DNS server, keep in mind that it is very important to secure your DNS server properly. A cracked public DNS server can be used to redirect traffic to any server the bad guys choose. So, if you are using that server, you are in danger of being presented with sites that are not the sites you think they are.

Configuring Linux as a proxy server

A proxy server provides a means of restricting network traffic from a private network to a public one, such as the Internet. Such servers provide an excellent way to lock down a computer lab at a school or restrict websites that employees can visit from work.

By physically setting up Linux as a router but configuring it as a proxy server, all of the systems on your home or business network can be configured to access the Internet using only certain protocols and only after you filter the traffic.

Using the Squid Proxy Server, which comes with most Linux systems (`squid` package in Fedora and RHEL), you can enable the system to accept requests to web servers (HTTP and HTTPS), file servers (FTP), and other protocols. You can restrict which systems can use your proxy server (by hostname or IP address) and even limit which sites they can visit (by specific address, range of addresses, hostname, or domain names).

Configuring a squid proxy server can be as simple as installing the `squid` package, editing the `/etc/squid/squid.conf` file, and starting the `squid` service. The file comes with a recommended minimal configuration. However, you might want to define the hosts (based on IP address or name) that you want to allow to use the service. There are blacklists available with `squid` that allow you to deny access to whole sets of sites that might be inappropriate for children to visit.

Summary

Most network connections from a Linux desktop or laptop system can be made with little or no user intervention. If you use NetworkManager over a wired or wireless Ethernet connection, address and server information needed to start up can be automatically obtained from a DHCP server.

With NetworkManager's graphical interface, you can do some network configuration, if you like. You can set static IP addresses and select the name server and gateway computers to use. To do more manual and complex network configuration, consider working more directly with network configuration files.

Network configuration files in Linux can be used to set up more advanced features such as Ethernet channel bonding.

Beyond the basics of network connectivity in Linux, features are available that enable you to provide network infrastructure types of services. This chapter introduced services and features such as routing, DHCP, and DNS that you need to know when working with more advanced networking features in Linux.

With your networking configured, you can now begin configuring services to run over your networks. Chapter 15, “Starting and Stopping Services,” describes the tools that you need to enable, disable, start, stop, and check the status of the services that are configured for your Linux system.

Exercises

The exercises in this section help you to examine and change the network interfaces on your Linux system as well as understand how to configure more advanced networking features. Start these exercises on a Linux system that has an active network connection but that is *not* in the middle of some critical network activity.

I recommend that you do these exercises directly from your computer console (in other words, don’t ssh into the computer to do them). Some of the commands that you run may interrupt your network connectivity, and some of the configuration you do, if you make a mistake, can result in your computer being temporarily unavailable from the network.

There are often multiple ways to complete the tasks described in these exercises. If you are stuck, refer to the task solutions that are provided in Appendix B.

1. Use the desktop to check that NetworkManager has successfully started your network interface (wired or wireless) to the network. If it has not, then try to start your network interface.
2. Run a command to check the active network interfaces available on your computer.
3. Try to contact `google.com` from the command line in a way that ensures that DNS is working properly.
4. Run a command to check the routes being used to communicate outside of your local network.
5. Trace the route being taken to connect to `google.com`.
6. View the network activity of your Linux system from the Cockpit web user interface.
7. Create a host entry that allows you to communicate with your local host system using the name `myownhost`.

8. Determine the addresses of the DNS name servers that are being used to resolve hostnames to IP addresses on your system, then check which is queried from your system to find the IP address for `google.com`.
9. Create a custom route that directs traffic destined for the 192.168.99.0/ 255.255.255.0 network to some IP address on your local network, such as 192.168.0.5 (first ensuring that the 192.168.99 network is not being used at your location).
10. Check to see if your system has been configured to allow IPv4 packets to be routed between network interfaces on your system.